

Space and Time Efficient Parallel Graph Decomposition, Clustering, and Diameter Approximation

Matteo Ceccarello

Department of Information Engineering
University of Padova
Padova, Italy
ceccarel@dei.unipd.it

Andrea Pietracaprina

Department of Information Engineering
University of Padova
Padova, Italy
capri@dei.unipd.it

Geppino Pucci

Department of Information Engineering
University of Padova
Padova, Italy
geppo@dei.unipd.it

Eli Upfal

Department of Computer Science
Brown University
Providence, RI, USA
eli_upfal@brown.edu

Abstract

We develop a novel parallel decomposition strategy for unweighted, undirected graphs, based on growing disjoint connected clusters from batches of centers progressively selected from yet uncovered nodes. With respect to similar previous decompositions, our strategy exercises a tighter control on both the number of clusters and their maximum radius.

We present two important applications of our parallel graph decomposition: (1) k -center clustering approximation; and (2) diameter approximation. In both cases, we obtain algorithms which feature a polylogarithmic approximation factor and are amenable to a distributed implementation that is geared for massive (long-diameter) graphs. The total space needed for the computation is linear in the problem size, and the parallel depth is substantially sublinear in the diameter for graphs with low doubling dimension. To the best of our knowledge, ours are the first parallel approximations for these problems which achieve sub-diameter parallel time, for a relevant class of graphs, using only linear space. Besides the theoretical guarantees, our algorithms allow for a very simple implementation on clustered architectures: we report on extensive experiments which demonstrate their effectiveness and efficiency on large graphs as compared to alternative known approaches.

1 Introduction

Graph analytics is rapidly establishing itself as a major discovery tool in such diverse application domains as road systems, social networks, natural language processing, biological pattern discovery, cybersecurity, and more. Graph analytics tasks for big data networks are typically run on distributed architectures such as clusters of loosely-coupled commodity servers, where the challenge is to minimize communication overhead between processors, while each processor can store only a small fraction of the entire network. A number of computational models proposed in recent years [14, 18, 24] provide excellent rigorous frameworks for studying algorithms for massive data, subject to these constraints. Under this new computational paradigm, state-of-the-art graph algorithms often do not scale up efficiently to process massive instances, since they either require superlinear memory or exhibit long critical paths resulting in a large number of communication rounds.

In this work we focus on *graph decomposition*, which is a fundamental primitive for graph analytics as well as for several other application contexts, especially in distributed settings, where decompositions are often at the base of efficient parallel solutions. We develop an efficient parallel decomposition algorithm for partitioning the nodes of an unweighted, undirected graph into disjoint, internally connected *clusters*, which is able to control the maximum radius of the clusters (the maximum distance of a node in a cluster to the cluster's center). Similarly to other known decomposition approaches, our algorithm grows clusters from several batches of centers which are progressively

selected from the uncovered nodes. However, rather than fixing the radius of each grown cluster *a priori*, or randomly delaying the activation of the centers, as in previous works, we activate a new batch of centers every time that the number of uncovered nodes halves, while continuing growing the clusters of previously activated centers. The idea behind such a strategy is to force more clusters to grow in poorly connected regions of the graph while keeping both the total number of clusters and the maximum cluster radius under control.

We demonstrate the quality and utility of our decomposition strategy by applying it to the solution of two extensively studied problems, metric k -center clustering and diameter approximation, for which we obtain space and time efficient parallel algorithms.

In the *metric k -center* clustering problem [13, 15] the goal is to partition an undirected graph into k clusters so that the maximum radius of the clusters is minimized. The problem is NP-hard and we are therefore interested in efficient approximations. Given an n node unweighted undirected graph, building on our parallel graph decomposition method, we obtain a randomized $O(\log^3 n)$ -approximation algorithm to the k -center problem. The algorithm can be implemented on the MapReduce (MR) model of [24] in a number of parallel rounds proportional to the maximum cluster radius using overall linear space, as long as each processing node is provided with $\Omega(n^\epsilon)$ local space, for any constant $\epsilon > 0$. In order to derive a more explicit bound on the parallel complexity, we analyze the maximum cluster radius, hence the number of rounds, as a function of the doubling dimension of the graph (see Definition 2), showing that for a graph of diameter Δ and doubling dimension $b > 0$ the algorithm can provide a decomposition into k clusters with maximum cluster radius $\tilde{O}(\lceil \Delta/k^{1/b} \rceil)$.

Next, we apply our graph decomposition strategy to a challenging problem in the context of graph analytics, namely, the approximation of the graph diameter, a global property of a graph, in a number of parallel rounds which is substantially less than the diameter itself, and using linear global space and local memory at each processor sufficient to store only a small fraction of the graph. We remark that known parallel approaches to estimating the diameter of a graph either require up to quadratic space (e.g., using transitive closure) or require a number of parallel rounds which is inherently linear in the diameter (e.g., using straightforward parallelization of breadth-first search or neighborhood function estimations).

To estimate the diameter of a graph G , we first compute a decomposition of G of suitable granularity with our novel algorithm, and then we estimate the diameter through the diameter of the *quotient graph*, that is, the graph whose nodes correspond to the clusters and whose edges connect clusters containing nodes which are adjacent in G . The granularity is chosen so that the size of the quotient graph is small enough so that its diameter can be computed using limited local memory and very few communication rounds.

We show that on any unweighted, undirected connected graph G with n nodes and m edges, our algorithm returns an upper bound to its diameter which is a factor $O(\log^3 n)$ away from the true value, with high probability. The algorithm can be implemented on the aforementioned MR model using overall linear space and $\Omega(n^\epsilon)$ local space, with $\epsilon > 0$, in a number of parallel rounds which is $\tilde{O}(\lceil \Delta/(\max\{m^{1/3}n^{\epsilon/6}, n^{\epsilon'}\})^{1/b} \rceil)$ where b is the doubling dimension of the graph and ϵ' is any constant less than ϵ . Observe that for graphs with small (e.g., constant) doubling dimension, which arise in important practical realms [17], the number of rounds can be made asymptotically much smaller than the diameter if sufficient yet sublinear local memory is available. While a similar approach for diameter estimation has been used in the past in the external-memory setting (see Section 2), the algorithm presented here, to the best of our knowledge, is the first linear-space distributed algorithm for the problem requiring a number of parallel rounds which is sublinear in the diameter.

A very desirable feature of our algorithms is that they lend themselves to efficient practical implementations. We report on an extensive set of experiments conducted on a number of large graphs. A first batch of experiments shows the effectiveness of our decomposition strategy in minimizing the maximum cluster radius, compared against the recent parallel decomposition strategy of [22], while a second batch shows that the approximation obtained by our diameter approximation algorithm is in fact much smaller than the asymptotic bound, even for graphs of unknown doubling dimension (less than twice the diameter in all tested cases) and that the algorithm's performance compares very favorably to the one exhibited by direct competitors such as breadth-first search and the (almost exact) diameter estimation algorithm HADI [16]. For graphs of very large diameter, the speed-up of our algorithm can be of orders of magnitude.

The rest of the paper is organized as follows. Section 2 summarizes relevant previous work on graph decomposition, k -center clustering, and diameter estimation. Section 3 presents our novel decomposition and discusses how it can be employed to approximate the k -center problem. Section 4 presents our decomposition-based algorithm for diameter approximation. Section 5 analyzes our strategies in the MR model of [24]. Section 6 reports on the experimental results, and, finally, Section 7 offers some concluding remarks.

2 Previous work

Parallel clustering algorithms relevant to this work have been studied in [5, 8, 22]. In [8] the notion of (β, W) -cover for a weighted graph G is introduced, which is essentially a decomposition of the graph into nondisjoint clusters, where each node is allowed to belong to $O(\beta n^{1/\beta} \log n)$ distinct clusters and for any two nodes at weighted distance at most W in the graph, there is a cluster containing both. A (β, W) -cover is obtained by growing clusters of decreasing radii from successive batches of centers. The algorithm presented in [5] is similar but returns disjoint clusters and guarantees a bound on the average number of edges between clusters. In [22] an alternative clustering algorithm is proposed which assigns to each node $u \in V$ a random time shift δ_u , taken from an exponential distribution with parameter β , and grows a cluster centered at u starting at time $\delta_{\max} - \delta_u$, where δ_{\max} is the maximum shift, unless by that time node u has been already covered by some other cluster. The authors show that in this fashion the graph is partitioned into clusters of maximum radius $O((\log n)/\beta)$, with high probability, while the average number of edges between clusters, hence the size of the quotient graph, is at most $O(\beta m)$. None of the above clustering approaches guarantees that the maximum radius of the returned clusters is (close to) minimum with respect to all possible decompositions of the graph featuring the same number of clusters.

The related *metric k -center* optimization problem requires that given a set V of n points in a metric space, a subset $M \subset V$ of k points be found so to minimize the maximum distance between any $v \in V$ and M . The problem is NP-hard even if distances satisfy the triangle inequality [28], but polynomial-time 2-approximation sequential algorithms are known [13, 15]. Recently, a constant-approximation MapReduce algorithm was developed in [12] under the assumption that the distances among all $\Theta(n^2)$ pairs of points are given in input. The problem remains NP-hard even if V represents the set of nodes of an undirected connected graph G , and the distance between two nodes is the length of the shortest path between them. To the best of our knowledge, no low-depth linear-space parallel strategy that yields a provably good approximation for this important graph variant is known in the literature.

In recent years, efficient sequential algorithms for estimating the diameter of very large graphs have been devised, which avoid the costly computation of all-pairs shortest paths or the memory-inefficient transitive closure by performing a limited number of Breadth-First Searches (BFS) from suitably selected source nodes [7, 10]. Unfortunately, due to the inherent difficulty of parallelizing BFS [19, 27] these approaches do not appear amenable to efficient low-depth parallel implementations. External-memory algorithms for diameter estimation which employ a clustering-based strategy similar to ours have been recently proposed in [3, 21]. The algorithm by [21] basically selects k centers at random and grows disjoint clusters around the centers until the whole graph is covered. The author shows that the diameter of the original graph can be approximated within a multiplicative factor of $\Theta(\sqrt{k} \log n)$, with high probability, by computing the diameter on the quotient graph associated with the clustering with suitable edge weights. In [3] a recursive implementation of this strategy is evaluated experimentally. This approximation ratio is competitive with our result only for polylogarithmic values of k . However, observe that for such small values of k the radius of the k clusters must be within a small (polylogarithmic) factor of the graph diameter, and thus the parallel number of rounds cannot be substantially sublinear in the diameter itself.

Efficient PRAM algorithms for approximating shortest path distances between given pairs of nodes are given in [8, 9]. For sparse graphs with $m \in \Theta(n)$, these algorithms feature $O(n^\delta)$ depth, for any fixed constant $\delta \in (0, 1)$, but incur a polylogarithmic space blow-up due to the use of the (β, W) -covers mentioned above. The algorithms are rather involved and communication intensive, hence, while theoretically efficient, in practice they may run slowly when implemented on distributed-memory clusters of loosely-coupled servers, where communication overhead is typically high. Moreover, their depth is not directly related to the graph diameter.

In [23], an efficient algorithm, called ANF, is devised to tightly approximate the *neighborhood function* of a graph G , which, for every $t \geq 0$, gives the number of pairs of nodes at distance at most t in G , and, therefore, it can be used to estimate the diameter. On a connected graph G of diameter Δ , ANF executes Δ iterations and maintains at each node v a suitable succinct data structure to approximate, at the end of each Iteration t , the number of nodes at distance at most t from v . A MapReduce implementation of ANF, called HADI, has been devised in [16] using Apache Hadoop. Little experimental evidence of HADI's performance on large benchmark graphs is available. However, as confirmed by our experiments (see Section 6), for large-diameter graphs HADI's strategy, even if implemented using faster engines than Hadoop, runs very slowly because of the large number of rounds and the high communication volume. A very fast, multithreaded version of ANF, called HyperANF, has been devised in [6] for expensive tightly-coupled multiprocessors with large shared memories, which are not the architectures targeted by our work.

3 Clustering algorithm

Let $G = (V, E)$ be an undirected connected graph with $n = |V|$ nodes and $m = |E|$ edges. For any two nodes $u, v \in V$ let $\text{dist}(u, v)$ denote the number of edges in the shortest path between u and v in G . Also, for any $u \in V$ and $X \subseteq V$, let $\text{dist}(u, X)$ denote the minimum distance between u and a node of X . We now present an algorithm (**CLUSTER**) that partitions V into disjoint clusters around suitably selected nodes called *centers*, so that the radius of each cluster, defined as the maximum distance of a cluster node from the center, is small. As in [5, 8, 22], our algorithm activates batches of centers progressively, so to allow more clusters to cover sparser regions of the graph. However, unlike those previous works, we can show that our algorithm minimizes the maximum cluster radius, within a polylogarithmic factor, a property that later will turn out crucial for the efficiency of the diameter-approximation algorithm. A parameter $\tau > 0$ is used to control the size of each batch of activated centers. When two or more clusters attempt to cover a node concurrently, only one of them, arbitrarily chosen, succeeds, so to maintain clusters disjoint. The algorithm's pseudocode is given below¹.

Algorithm 1: **CLUSTER**(τ)

```

begin
   $C \leftarrow \emptyset$  // (current set of clusters)
   $V' \leftarrow \emptyset$  // (nodes covered by current set of clusters)
  while  $|V - V'| \geq 8\tau \log n$  do
    Select each node of  $V - V'$  as a new center
    independently with probability  $4\tau \log n / |V - V'|$ 
    Add to  $C$  the set of singleton clusters centered at the
    selected nodes
    In parallel grow all clusters of  $C$  disjointly until
     $\geq |V - V'|/2$  new nodes are covered
     $V' \leftarrow$  nodes covered by the clusters in  $C$ 
  end
  return  $C \cup \{\text{singleton clusters centered at the nodes of } V - V'\}$ 
end

```

We define a crucial benchmark for analyzing each iteration of the algorithm.

Definition 1. Let k be an integer, with $1 \leq k \leq n$, and let $V' \subseteq V$ be a subset of at most $n - k$ nodes. We define

$$r(G, V', k) = \min\{r : \exists U \subseteq V - V' \text{ s.t. } |U| = k \\ \wedge \forall w \in V - V' \text{ dist}(w, V' \cup U) \leq r\}.$$

Suppose that we have partially grown some clusters covering a subset $V' \subset V$. We know that by continuing to grow these clusters plus k new clusters centered in uncovered nodes, $r = r(G, V', k)$ growing steps are necessary to cover the whole graph. We have:

Theorem 1. For any integer $\tau > 0$, with high probability, **CLUSTER**(τ) computes a partition of V into $O(\tau \cdot \log^2 n)$ disjoint clusters of maximum radius

$$R_{\text{ALG}} = O\left(\sum_{i=1}^{\ell} R(i, \tau)\right)$$

where $\ell = \lceil \log(n / (8\tau \log n)) \rceil$, and

$$R(i, \tau) = \max\{r(G, V', \tau) : V' \subset V \wedge |V - V'| = n/2^{i-1}\}$$

for $1 \leq i \leq \ell$.

Proof. The bound on the number of clusters follows by observing that the number of clusters added in each iteration is a binomial random variable with expectation $4\tau \log n$, and that at most $\ell = O(\log n)$ iterations are executed overall.

¹Unless explicitly indicated, the base of all logarithms is 2.

As for the upper bound on R_{ALG} , it is sufficient to show that in the i th iteration, with $i \geq 1$, the radius of each cluster (new or old) grows by $O(R(i, \tau))$, with high probability. Let V_i be the set of nodes that at the beginning of Iteration i are already covered by the existing clusters. By construction, we have that $|V_1| = 0$ and, for $i > 1$, $|V_i| \geq \sum_{j=1}^{i-1} n/2^j$. Hence, $|V - V_i| \leq n/2^{i-1}$. Let $R_i = r(G, V_i, \tau)$. It is easy to verify that

$$\begin{aligned} R_i &\leq \max_{V' \subset V, |V-V'|=|V-V_i|} r(G, V', \tau) \\ &\leq \max_{V' \subset V, |V-V'|=n/2^{i-1}} r(G, V', \tau) \\ &= R(i, \tau). \end{aligned}$$

By definition of $r(G, V_i, \tau)$ we know that there must exist τ nodes, say $u_1, u_2, \dots, u_\tau \in V - V_i$, such that each node of $V - V_i$ is at distance at most R_i from either V_i or one of these nodes. Let us consider the partition

$$V - V_i = B_0 \cup B_1 \cup \dots \cup B_\tau$$

where B_0 is the set of nodes of $V - V_i$ which are closer to V_i than to any of the u_j 's, while B_j is the set of nodes of $V - V_i$ which are closer to u_j than to V_i or to any other $u_{j'}$. Let

$$J = \{j \geq 1 : |B_j| \leq |V - V_i|/(2\tau)\}$$

and note that

$$\sum_{j \in J} |B_j| \leq \tau(|V - V_i|/(2\tau)) \leq |V - V_i|/2$$

Therefore, we have that $\sum_{j \notin J} |B_j| \geq |V - V_i|/2$. Since $|V - V_i| \geq 8\tau \log n$, it is easy to see that for any $j \geq 1$ and $j \notin J$, in the i th iteration a new center will be chosen from B_j with probability at least $1 - 1/n^2$. Hence, by the union bound, we conclude that a new center will fall in every B_j with $j \geq 1$ and $j \notin J$, with probability at least $1 - \tau/n^2$. When this event occurs, $R_i \leq R(i, \tau)$ cluster growing steps will be sufficient to reach half of the nodes of $V - V_i$ (namely, the nodes belonging to $B_0 \cup (\cup_{j \notin J} B_j)$). The theorem follows by applying the union bound over the $\ell \leq \log n$ iterations. \square

An important issue, which is crucial to assess the efficiency of the diameter approximation algorithm discussed in the next section, is to establish how much smaller is the maximum radius R_{ALG} of the clusters returned by `CLUSTER` with respect to the graph diameter Δ , which is an obvious upper bound to R_{ALG} . Our analysis will express the relation between R_{ALG} and Δ as a function of the *doubling dimension* of the graph, a concept that a number of recent works have shown to be useful in relating algorithms' performance to graph properties [2].

Definition 2. Consider an undirected graph $G = (V, E)$. The ball of radius R centered at node v is the set of nodes at distance at most R from v . Also, the doubling dimension of G is the smallest integer $b > 0$ such that for any $R > 0$, any ball of radius $2R$ can be covered by at most 2^b balls of radius R .

The following lemma provides an upper bound on R_{ALG} in terms of the doubling dimension and of the diameter of the graph G .

Lemma 1. Let G be a connected n -node graph with doubling dimension b and diameter Δ . For $\tau \in O(n/\log^2 n)$, with high probability, `CLUSTER`(τ) computes a partition of V into $O(\tau \cdot \log^2 n)$ disjoint clusters of maximum radius

$$R_{\text{ALG}} = O\left(\left\lceil \frac{\Delta}{\tau^{1/b}} \right\rceil \log n\right).$$

Proof. Let $R_{\text{OPT}}(\tau)$ be the smallest maximum radius achievable by any decomposition into τ clusters. It is easy to see that each $R(i, \tau)$ is a lower bound to $R_{\text{OPT}}(\tau)$, whence $R_{\text{ALG}} = O\left(\sum_{i=1}^{\ell} R(i, \tau)\right) = O(R_{\text{OPT}}(\tau) \log n)$. By iterating the definition of doubling dimension starting from a single ball of radius Δ containing the whole graph, one can easily argue that G can be decomposed into (at most) τ disjoint clusters of radius $R = O(\lceil \Delta/\tau^{1/b} \rceil)$. The bound on R_{ALG} follows since $R = \Omega(R_{\text{OPT}}(\tau))$. \square

Observe that for graphs with diameter $\Delta = \omega(\log n)$ and low (e.g., constant) doubling dimension, R_{ALG} becomes $o(\Delta)$ when τ is large enough. Indeed, some experimental work [17] reported that, in practice, big data networks of interest have low doubling dimension. Also, for applications such as the diameter estimation discussed in the next section, it is conceivable that parameter τ be made as large as n^ϵ , for some constant $\epsilon > 0$. In fact, the gap between the graph diameter and R_{ALG} can be even more substantial for irregular graphs where highly-connected regions and sparsely-connected ones coexist. For example, let G consist of a constant-degree expander of $n - \sqrt{n}$ nodes attached to a path of \sqrt{n} nodes, and set $\tau = \sqrt{n}$. It is easy to see that $R(i, \tau) = O(\text{poly}(\log n))$, for $i \geq 1$. Hence, $\text{CLUSTER}(\tau)$ returns $\sqrt{n} \log^2 n$ clusters of maximum radius $R_{\text{ALG}} = O(\text{poly}(\log n))$, which is exponentially smaller than the $\Omega(\sqrt{n})$ graph diameter.

3.1 Approximation to k -center

Algorithm CLUSTER can be employed to compute an approximate solution to the k -center problem, defined as follows. Given an undirected connected graph $G = (V, E)$ with unit edge weights, a set $M \subseteq V$ of k centers is sought which minimizes the maximum distance $R_{\text{OPT}}(k)$ in G of any node $v \in V$ from M . As mentioned in Section 2 this problem is NP-hard. The theorem below states our approximation result.

Theorem 2. *For $k = \Omega(\log^2 n)$, algorithm CLUSTER can be employed to yield a $O(\log^3 n)$ -approximation to the k -center problem with unit edge weights, with high probability.*

Proof. Fix $\tau = \Theta(k / \log^2 n)$ so that our algorithm returns at most k clusters with high probability, and let M be the set of centers of the returned clusters. Without loss of generality, we assume that M contains exactly k nodes (in case $|M| < k$, we can add $k - |M|$ arbitrary nodes to M , which will not increase the value of objective function). Let R_{ALG} be the maximum radius of the clusters returned by our algorithm. As proved in Lemma 1, we have that, with high probability $R_{\text{ALG}} = O(R_{\text{OPT}}(\tau) \log n)$. We conclude the proof by arguing that $R_{\text{OPT}}(\tau) = O(R_{\text{OPT}}(k) \log^2 n)$. Consider the optimal solution to the k -center problem on the graph, and the associated clustering of radius $R_{\text{OPT}}(k)$. Let T be a spanning tree of the quotient graph associated with this clustering. It is easy to see that T can be decomposed into τ subtrees of height $O(\log^2 n)$ each. Merge the clusters associated with the nodes of each such subtree into one cluster and pick any node as center of the merged cluster. It is easy to see that every node in the graph is at distance $D = O(R_{\text{OPT}}(k) \log^2 n)$ from one of the picked nodes. Since $D \geq R_{\text{OPT}}(\tau)$, we conclude that $R_{\text{OPT}}(\tau) = O(R_{\text{OPT}}(k) \log^2 n)$, and the theorem follows. \square

3.2 Extension to disconnected graphs

Let G be an n -node graph with $h > 1$ connected components. It is easy to see that for any $\tau \geq h$, algorithm $\text{CLUSTER}(\tau)$ works correctly with the same guarantees stated in Theorem 1. Also, observe that for $k \geq h$, the k -center problem still admits a solution with noninfinite radius. Given $k \geq h$, we can still get a $O(\log^3 n)$ -approximation to k -center on G as follows. If $k = \Omega(h \log^2 n)$ we simply run $\text{CLUSTER}(\tau)$ with $\tau = \Theta(k / \log^2 n)$ as before. If instead $h \leq k = o(h \log^2 n)$ we run $\text{CLUSTER}(h)$ and then reduce the number of clusters $W = O(h \log^2 n)$ returned by the algorithm to k by using the merging technique described in the proof of Theorem 2. It is easy to show that the approximation ratio is still $O(\log^3 n)$.

4 Diameter estimation

Let G be an n -node connected graph. As in [21], we approximate the diameter of G through the diameter of the quotient graph associated with a suitable clustering of G . For the distributed implementation discussed in the next section, the clustering will be made sufficiently coarse so that the diameter of the quotient graph can be computed on a single machine. In order to ensure a small approximation ratio, we need a refined clustering algorithm (CLUSTER2), whose pseudocode is given in Algorithm 2, which imposes a lower bound on the number of growing steps applied to each cluster, where such a number is precomputed using the clustering algorithm from Section 3.

Let $\tau > 0$ be an integral parameter. We have:

Lemma 2. *For any integer $\tau > 0$, with high probability algorithm $\text{CLUSTER2}(\tau)$ computes a partition of V into $O(\tau \log^4 n)$ clusters of radius $R_{\text{ALG2}} \leq 2R_{\text{ALG}} \log n$, where R_{ALG} is the maximum radius of a cluster returned by $\text{CLUSTER}(\tau)$.*

Algorithm 2: CLUSTER2(τ)

```
begin
  Run CLUSTER( $\tau$ ) and let  $R_{\text{ALG}}$  be the maximum radius of
  the returned clusters
   $C \leftarrow \emptyset$  // (current set of clusters)
   $V' \leftarrow \emptyset$  // (nodes covered by current set of clusters)
  for  $i \leftarrow 1$  to  $\log n$  do
    Select each node of  $V - V'$  as a new center
    independently with probability  $2^i/n$ 
    Add to  $C$  the set of singleton clusters centered at the
    selected nodes
    In parallel grow all clusters of  $C$  disjointly for
     $2R_{\text{ALG}}$  steps
     $V' \leftarrow$  nodes covered by the clusters in  $C$ 
  end
  return  $C$ 
end
```

Proof. The bound on $R_{\text{ALG}2}$ is immediate. Let W be the number of clusters returned by the execution of CLUSTER(τ) within CLUSTER2(τ). By Theorem 1, we have that $W = O(\tau \log^2 n)$, with high probability. In what follows, we condition on this event. For $\gamma = 4/\log_2 e$, define H as the smallest integer such that $2^H/n \geq (\gamma W \log n)/n$, and let $t = \log n - H$. For $0 \leq i \leq t$, define the event $E_i =$ "at the end of Iteration $H + i$ of the **for** loop, at most $n/2^i$ nodes are still uncovered". We now prove that the event $\cap_{i=0}^t E_i$ occurs with high probability. Observe that

$$\begin{aligned} \Pr\left(\cap_{i=0}^t E_i\right) &= \Pr(E_0) \prod_{i=1}^t \Pr(E_i | E_0 \cap \dots \cap E_{i-1}) \\ &= \prod_{i=1}^t \Pr(E_i | E_0 \cap \dots \cap E_{i-1}) \end{aligned}$$

since E_0 clearly holds with probability one. Consider an arbitrary i , with $0 < i \leq t$, and assume that $E_0 \cap \dots \cap E_{i-1}$ holds. We prove that E_i holds with high probability. Let V_i be the set of nodes already covered at the beginning of Iteration $H + i$. Since E_{i-1} holds, we have that $|V - V_i| \leq n/2^{i-1}$. Clearly, if $|V - V_i| \leq n/2^i$ then E_i must hold with probability one. Thus, we consider only the case

$$\frac{n}{2^i} < |V - V_i| \leq \frac{n}{2^{i-1}}$$

Let $R_i = r(G, V_i, W)$ and observe that since R_{ALG} is the maximum radius of a partition of G into W clusters, we have that $R_{\text{ALG}} \geq R_i$. By the definition of R_i , there exist W nodes, say $u_1, u_2, \dots, u_W \in V - V_i$, such that each node of $V - V_i$ is at distance at most R_i from either V_i or one of these nodes. Let us consider the partition

$$V - V_i = B_0 \cup B_1 \cup \dots \cup B_W$$

where B_0 is the set of nodes of $V - V_i$ which are closer to V_i than to any of the u_j 's, while B_j is the set of nodes of $V - V_i$ which are closer to u_j than to V_i or to any other $u_{j'}$ (with ties broken arbitrarily). Let

$$J = \{j \geq 1 : |B_j| \geq |V - V_i|/(2W)\}$$

It is easy to see that $|B_0| + \sum_{j \in J} |B_j| \geq |V - V_i|/2$. Since we assumed that $|V - V_i| > n/2^i$, we have that for every B_j with $j \in J$,

$$|B_j| \geq \frac{|V - V_i|}{2W} \geq \frac{n\gamma \log n}{2^{H+i+1}}$$

As a consequence, since $\gamma = 4/\log_2 e$, a new center will be chosen from B_j in Iteration $H + i$ with probability at least $1 - 1/n^2$. By applying the union bound we conclude that in Iteration $H + i$ a new center will fall in every B_j with $j \in J$, and thus the number of uncovered nodes will at least halve, with probability at least $1 - W/n^2 \geq 1 - 1/n$.

By multiplying the probabilities of the $O(\log n)$ conditioned events, we conclude that event $\cap_{i=0}^t E_i$ occurs with high probability. Finally, one can easily show that, with high probability, in the first H iterations, $O(W \log n)$ clusters are added and, by conditioning on $\cap_{i=0}^t E_i$, at the beginning of each Iteration $H + i$, $1 \leq i \leq t$, $O(W \log n)$ new clusters are added to C , for a total of $O(W \log^2 n) = O(\tau \log^4 n)$ clusters. \square

Suppose we run CLUSTER2 on a graph G , for some $\tau \in O(n/\log^4 n)$, to obtain a set C of clusters of maximum radius R_{ALG2} . Let G_C denote the quotient graph associated with the clustering, where the nodes correspond to the clusters and there is an edge between two nodes if there is an edge of G whose endpoints belong to the two corresponding clusters. Let Δ_C be the diameter of G_C . We have:

Theorem 3. *If Δ is the true diameter of G , then $\Delta_C = O((\Delta/R_{\text{ALG}}) \log^2 n)$, with high probability.*

Proof. Let us fix an arbitrary pair of distinct nodes and an arbitrary shortest path π between them. We show that at most $O(\lceil |\pi|/R_{\text{ALG}} \rceil \log^2 n)$ clusters intersect π (i.e., contain nodes of π), with high probability. Divide π into segments of length R_{ALG} , and consider one such segment S . Clearly, all clusters containing nodes of S must have their centers belong to nodes at distance at most R_{ALG2} from S (i.e., distance at most R_{ALG2} from the closest node of S). Recall that $R_{\text{ALG2}} \leq 2R_{\text{ALG}} \log n$. For $1 \leq j \leq 2 \log n$, let $C(S, j)$ be the set of nodes whose distance from S is between $(j-1)R_{\text{ALG}}$ and jR_{ALG} , and observe that any cluster intersecting S must be centered at a node belonging to one of the $C(S, j)$'s. We claim that, with high probability, for any j , there are $O(\log n)$ clusters centered at nodes of $C(S, j)$ which may intersect S . Fix an index j , with $1 \leq j \leq 2 \log n$, and let i_j be the first iteration of the for loop of algorithm CLUSTER2 in which some center is selected from $C(S, j)$. It is easy to see that, due to the smooth growth of the center selection probabilities, the number of centers selected from $C(S, j)$ in Iteration i_j and in Iteration $i_j + 1$ is $O(\log n)$, with high probability. Consider now a center v (if any) selected from $C(S, j)$ in some Iteration $i > i_j + 1$. In order to reach S , the cluster centered at v must grow for at least $(j-1)R_{\text{ALG}}$ steps. However, since in each iteration active clusters grow by $2R_{\text{ALG}}$ steps, by the time the cluster centered at v reaches S , the nodes of S have already been reached and totally covered by clusters whose centers have been selected from $C(S, j)$ in Iterations i_j and $i_j + 1$ or, possibly, by some other clusters centered outside $C(S, j)$. In conclusion, we have that the nodes of segment S will belong to $O(\log^2 n)$ clusters, with high probability. The theorem follows by applying the union bound over all segments of π , and over all pairs of nodes in G . \square

Let $\Delta' = 2R_{\text{ALG2}} \cdot (\Delta_C + 1) + \Delta_C$. It is easy to see that $\Delta_C \leq \Delta \leq \Delta'$. Moreover, since $R_{\text{ALG2}} \leq 2R_{\text{ALG}} \log n$ and $R_{\text{ALG2}} = O(\Delta)$, we have from Theorem 3 that $\Delta' = O(\Delta \log^3 n)$. The following corollary is immediate.

Corollary 1. *Let G be an n -node connected graph with diameter Δ . Then, the clustering returned by CLUSTER2 can be used to compute two values Δ_C, Δ' such that $\Delta_C \leq \Delta \leq \Delta' = O(\Delta \log^3 n)$, with high probability.*

In order to get a tighter approximation, as in [21], after the clustering we can compute the diameter Δ'_C of the following weighted instance of the quotient graph $G_C = (V_C, E_C)$. Specifically, we assign to each edge $(u, v) \in E_C$ a weight equal to the length of the shortest path in G that connects the two clusters associated with u and v and comprises only nodes of these two clusters. It is easy to see that $\Delta'' = 2R_{\text{ALG2}} + \Delta'_C$ is an upper bound to the diameter Δ of G , and $\Delta'' \leq \Delta'$.

It is important to remark that while in [21] the approximation factor for the diameter is proportional to the square root of the number of clusters, with our improved clustering strategy the approximation factor becomes independent of this quantity, a fact that will also be confirmed by the experiments. As we will see in the next section, the number of clusters, hence the size of the quotient graph, can be suitably chosen to reduce the complexity of the algorithm, based on the memory resources.

As a final remark, we observe that the proof of Theorem 3 shows that for any two nodes u, v in G their distance $d(u, v)$ can be upper bounded by a value $d'(u, v) = O(d(u, v) \log^3 n + R_{\text{ALG2}})$. As a consequence, by running CLUSTER2(τ) with $\tau = O(\sqrt{n}/\log^4 n)$ and computing the $O(n)$ -size all-pairs shortest-path matrix of the (weighted) quotient graph G_C we can obtain a linear-space distance oracle for G featuring the aforementioned approximation quality, which is polylogarithmic for farther away nodes (i.e., nodes at distance $\Omega(R_{\text{ALG2}})$).

5 Distributed implementation and performance analysis

We now describe and analyze a distributed implementation of the clustering and diameter-approximation algorithms devised in the previous sections, using the MapReduce (MR) model introduced in [24]. The MR model provides

a rigorous computational framework based on the popular MapReduce paradigm [11], which is suitable for large-scale data processing on clusters of loosely-coupled commodity servers. Similar models have been recently proposed in [14, 18]. An MR algorithm executes as a sequence of *rounds* where, in a round, a multiset X of key-value pairs is transformed into a new multiset Y of pairs by applying a given reducer function (simply called *reducer* in the rest of the paper) independently to each subset of pairs of X having the same key. The model features two parameters M_G and M_L , where M_G is the maximum amount of global memory available to the computation, and M_L is the maximum amount of local memory available to each reducer. We use $\text{MR}(M_G, M_L)$ to denote a given instance of the model. The complexity of an $\text{MR}(M_G, M_L)$ algorithm is defined as the number of rounds executed in the worst case, and it is expressed as a function of the input size and of M_G and M_L . Considering that for big input instances local and global space are premium resources, the main aim of algorithm design on the model is to provide strategies exhibiting good space-round tradeoffs for large ranges of the parameter values.

The following facts are proved in [14, 24].

Fact 1. *The sorting and (segmented) prefix-sum primitives for inputs of size n can be performed in $O(\log_{M_L} n)$ rounds in $\text{MR}(M_G, M_L)$ with $M_G = \Theta(n)$.*

Fact 2. *Two $\ell \times \ell$ -matrices can be multiplied in $O(\log_{M_L} n + \ell^3/(M_G \sqrt{M_L}))$ rounds in $\text{MR}(M_G, M_L)$.*

We can implement the sequence of cluster-growing steps embodied in the main loops of `CLUSTER` and `CLUSTER2` as a progressive shrinking of the original graph, by maintaining clusters coalesced into single nodes and updating the adjacencies accordingly. Each cluster-growing step requires a constant number of sorting and (segmented) prefix operations on the collection of edges. Moreover, the assignment of the original graph nodes to clusters can be easily maintained with constant extra overhead. By using Fact 1, we can easily derive the following result.

Lemma 3. *`CLUSTER` (resp., `CLUSTER2`) can be implemented in the $\text{MR}(M_G, M_L)$ model so that, when invoked on a graph G with n nodes and m edges, it requires $O(R \log_{M_L} m)$ rounds, where R is the total number of cluster-growing steps performed by the algorithm. In particular, if $M_L = \Omega(n^\epsilon)$, for some constant $\epsilon > 0$, the number of rounds becomes $O(R)$.*

The diameter-approximation algorithm can be implemented in the $\text{MR}(M_G, M_L)$ model by running `CLUSTER2`(τ) for a value of τ suitably chosen to allow the diameter of the quotient graph to be computed efficiently. The following theorem shows the space-round tradeoffs attainable when M_L is large enough.

Theorem 4. *Let G be a connected graph with n nodes, m edges, doubling dimension b and diameter Δ . Also, let $\epsilon' < \epsilon \in (0, 1)$ be two arbitrary constants. On the $\text{MR}(M_G, M_L)$ model, with $M_G = \Theta(m)$ and $M_L = \Theta(n^\epsilon)$, an upper bound $\Delta' = O(\Delta \log^3 n)$ to the diameter of G can be computed in*

$$O\left(\left\lceil \frac{\Delta \log^{4/b} n}{(\max\{m^{1/3} n^{\epsilon/6}, n^{\epsilon'}\})^{1/b}} \right\rceil \log^2 n\right)$$

rounds, with high probability.

Proof. Fix $\tau = \Theta(n^{\epsilon'}/\log^4 n)$ so that `CLUSTER2`(τ) returns $O(n^{\epsilon'})$ clusters with high probability. (In case the number of returned clusters is larger, we repeat the execution of `CLUSTER2`.) Let $G_C = (V_C, E_C)$ be quotient graph associated with the returned clustering. If $|E_C| \leq M_L$, we can compute the diameter of G_C in one round using a single reducer. Otherwise, by employing the sparsification technique presented in [4] we transform G_C into a new graph $G'_C = (V_C, E'_C)$ with $|E'_C| \leq M_L$, whose diameter is a factor at most $O(\epsilon'/(\epsilon - \epsilon')) = O(1)$ larger than the diameter of G_C . The sparsification technique requires a constant number of cluster growing steps similar in spirit to those described above, which can be realized through a constant number of prefix and sorting operations. Hence, the transformation can be implemented in $O(1)$ rounds in $\text{MR}(M_G, M_L)$. Once G'_C is obtained, its diameter and the resulting approximation Δ' to Δ can be computed in one round with a single reducer. Therefore, by combining the results of Lemmas 1, 2, and 3, we have that `CLUSTER`(τ) runs in $O(\lceil \Delta/\tau^{1/b} \rceil \log n)$ rounds, and `CLUSTER2`(τ) runs $O(\lceil \Delta/\tau^{1/b} \rceil \log^2 n)$ rounds. Hence, we have that the total number of rounds for computing Δ' is $O(\lceil \Delta \log^{4/b} n / n^{\epsilon'/b} \rceil \log^2 n)$. Alternatively, we can set $\tau = O(\min\{n, m^{1/3} n^{\epsilon/6} / \log^4 n\})$ so to obtain a quotient graph G_C with $|V_C| = \Theta(m^{1/3} n^{\epsilon/6})$ nodes. We can compute the diameter of the quotient graph

by repeated squaring of the adjacency matrix. By applying the result of Fact 2 with $\ell = |V_C|$ and observing that $|V_C|^3 = O(m \cdot n^{\epsilon/2}) = O(M_G \sqrt{M_L})$, we conclude that the computation of the quotient graph diameter requires only an extra logarithmic number of rounds. In this fashion, the total number of rounds for computing Δ' becomes $O\left(\lceil \Delta \log^{4/b} n / (m^{1/3} n^{\epsilon/6})^{1/b} \rceil \log^2 n\right)$. The theorem follows by noting that for both the above implementations, the quality of the approximation is ensured by Corollary 1. \square

We remark that while the upper bound on the approximation factor is independent of the doubling dimension of the graph, the round complexity is expressed as a function of it. This does not restrict the generality of the algorithm but allows us to show that for graphs with small doubling dimension, typically graphs with low expansion, the number of rounds can be made substantially smaller than the graph diameter and, in fact, this number decreases as more local memory is available for the reducers, still using linear global space. This feature represents the key computational advantage of our algorithm with respect to other linear-space algorithms, that, while yielding tighter approximations, require $\Omega(\Delta)$ rounds.

6 Experimental results

We tested our algorithms on a cluster of 16 hosts, each equipped with a 12 GB RAM and a 4-core I7 processor, connected by a 10 gigabit Ethernet network. The algorithms have been implemented using Apache Spark [26], a popular engine for distributed large-scale data processing. We performed tests on several large graphs whose main characteristics are reported in Table 1. The first graph is a symmetrization of a subgraph of the Twitter network obtained from the LAW website [20]. The next four graphs are from the Stanford Large Network Datasets Collection [25] and represent, respectively, the Livejournal social network and three road networks. The last graph is a synthetic 1000×1000 mesh, which has been included since its doubling dimension is known, unlike the other graphs, and constant ($b = 2$), hence it is an example of a graph where our algorithms are provably effective.

6.1 Experiments on the Clustering Algorithm

We compared the quality of the clustering returned by algorithm `CLUSTER` (see Section 3) against that of the clustering returned by the algorithm presented in [22] and reviewed in Section 2, which, for brevity, we call `MPX`. Recall that `CLUSTER` uses a parameter τ to control the number of clusters, while `MPX` uses (an exponential distribution of) parameter β to decide when nodes are possibly activated as cluster centers, hence indirectly controlling the number of clusters. Both algorithms aim at computing a decomposition of the graph into clusters of small radius, so we focused the experiments on comparing the maximum radius of the returned clusterings. However, since the minimum maximum radius attainable by any clustering is a nonincreasing function of the number of clusters, but neither algorithm is able to precisely fix such a number a priori, we structured the experiments as follows.

We aimed at decomposition granularities (i.e., number of clusters) which are roughly three orders of magnitude smaller than the number of nodes for small-diameter graphs, and roughly two orders of magnitude smaller than the number of nodes for large-diameter graphs. We ran `MPX` and `CLUSTER` setting their parameters β and τ so to obtain a granularity close enough to the desired one, and compared the maximum cluster radius obtained by the two algorithms. In order to be conservative, we gave `MPX` a slight advantage setting β so to always yield a comparable but larger number of clusters with respect to `CLUSTER`.

Table 2 shows the results of the experiments for the benchmark graphs. Each row reports the graph, and, for each algorithm, the number of nodes (n_C) and edges (m_C) of the quotient graph associated with the clustering, and the maximum cluster radius (r). The table provides a clear evidence that our algorithm is more effective in keeping the maximum cluster radius small, especially for graphs of large diameter. This is partly due to the fact that `MPX` starts growing only a few clusters, and before more cluster centers are activated the radius of the initial clusters is already grown large. On the other hand, `MPX` is often more effective in reducing the number of edges of the quotient graph, which is in fact the main objective of the `MPX` decomposition strategy. This is particularly evident for the first two graphs in the table, which represent social networks, hence feature low diameter and high expansion (thus, probably, high doubling dimension). In these cases, the few clusters initially grown by `MPX` are able to absorb entirely highly expanding components, thus resulting in a more drastic reduction of the edges.

6.2 Experiments on the Diameter-Approximation Algorithm

For the diameter approximation, we implemented a simplified version of the algorithm presented in Section 4, where, for efficiency, we used `CLUSTER` instead of `CLUSTER2`, thus avoiding repeating the clustering twice. Also, in order to get a tighter approximation, we computed the diameter of the weighted variant of the quotient graph as discussed at the end of Section 4. We performed three sets of experiments, which are discussed below.

The first set of experiments aimed at testing the quality of the diameter approximation provided by our algorithm. The results of the experiments are reported in Table 3. For each graph of Table 1 we estimated the diameter by running our algorithm with two clusterings of different granularities (dubbed *coarser* and *finer* clustering, respectively) reporting, in each case, the number of nodes (n_C) and edges (m_C) of the quotient graph G_C , the approximation Δ' and the true diameter Δ^2 . Since the quotient graphs turned out to be sufficiently sparse, the use of sparsification techniques mentioned in Section 5 was not needed. We observe that in all cases $\Delta'/\Delta < 2$ and, in fact, the approximation factor appears to decrease for sparse, long-diameter graphs. Also, we observe that, as implied by the theoretical results, the quality of the approximation does not seem to be influenced by the granularity of the clustering. Therefore, for very large graphs, or distributed platforms where individual machines are provided with small local memory, one can resort to a very coarse clustering in order to fit the whole quotient graph in one machine, and still obtain a good approximation to the diameter, at the expense, however, of an increased number of rounds, which are needed to compute the clustering.

With the second set of experiments, we assessed the time performance of our algorithm against two competitors: HADI [16], which was reviewed in Section 2 and provides a rather tight diameter (under)estimation; and Breadth First Search (BFS), which, as well known, can be employed to obtain an upper bound to the diameter within a factor two. HADI's original code, available from [1], was written for the Hadoop framework. Because of Hadoop's known large overhead, for fairness, we reimplemented HADI in Spark, with a performance gain of at least one order of magnitude. As for BFS, we implemented a simple and efficient version in Spark. Table 4 reports the running times and the diameter estimates obtained with the three algorithms where, for our algorithm, we used the finer clustering granularity adopted in the experiments reported in Table 3. The figures in the table clearly show that HADI, while yielding a very accurate estimate of the diameter, is much slower than our algorithm, by orders of magnitude for large-diameter graphs. This is due to the fact that HADI requires $\Theta(\Delta)$ rounds and in each round the communication volume is linear in the number of edges of the input graph. On the other hand BFS, whose approximation guarantee is similar to ours in practice, outperforms HADI and, as expected, is considerably slower than our algorithm on large-diameter graphs. Indeed, BFS still requires $\Theta(\Delta)$ rounds as HADI, but its aggregate communication volume (rather than the per round communication volume) is linear in the number of edges of the input graph.

As remarked in the discussion following Lemma 1, a desirable feature of our strategy is its capability to adapt to irregularities of the graph topology, which may have a larger impact on the performance of the other strategies. In order to provide experimental evidence of this phenomenon, our third set of experiments reports the running times of our algorithm and BFS on three variants of the two small-diameter graphs (livejournal and twitter) obtained by appending a chain of $c \cdot \Delta$ extra nodes to a randomly chosen node, with $c = 1, 2, 4, 6, 8, 10$, thus increasing the diameter accordingly, without substantially altering the overall structure of the base graph. The plots in Figure 1 clearly show that while the running time of our algorithm is basically unaltered by the modification, that of BFS grows linearly with c , as expected due to the strict dependence of the BFS number of rounds from the diameter. A similar behaviour is to be expected with HADI because of the same reason.

Putting it all together, the experiments support the theoretical analysis since they provide evidence that the main competitive advantages of our algorithm, which are evident in large-diameter graphs, are the linear aggregate communication volume (as in BFS) coupled with its ability to run in a number of rounds which can be substantially smaller than Δ .

7 Conclusions

We developed a novel parallel decomposition strategy for unweighted, undirected graphs which ensures a tighter control on both the number of clusters and their maximum radius, with respect to similar previous decompositions. We employed our decomposition to devise parallel polylogarithmic approximation algorithms for the k -center problem

²In fact, in some cases the “true diameter” reported in the table has been computed through approximate yet very accurate algorithms and may exhibit some small discrepancies with the actual value.

and for computing the graph diameter. The algorithms use only linear overall space and, for a relevant class of graphs (i.e., those of low doubling dimension), their parallel depth can be made substantially sublinear in the diameter as long as local memories at the processing nodes are sufficiently large but still asymptotically smaller than the graph size.

While the improvement of the approximation bounds and the parallel depth of our algorithms is a natural direction for further research, the extension of our findings to the realm of weighted graphs is another challenging and relevant open problem. We are currently exploring this latter issue and have devised a preliminary decomposition strategy that, together with the number clusters and their weighted radius, also controls their hop radius, which governs the parallel depth of the computation.

References

- [1] Project PEGASUS. www.cs.cmu.edu/~pegasus.
- [2] I. Abraham, S. Chechik, C. Gavoille, and D. Peleg. Forbidden-set distance labels for graphs of bounded doubling dimension. In *ACM PODC*, pages 192–200, 2010.
- [3] D. Ajwani, U. Meyer, and D. Veith. I/O-efficient hierarchical diameter approximation. In *ESA*, pages 72–83, 2012.
- [4] S. Baswana and S. Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007.
- [5] G. E. Blelloch, A. Gupta, I. Koutis, G. L. Miller, R. Peng, and K. Tangwongsan. Near linear-work parallel sdd solvers, low-diameter decomposition, and low-stretch subgraphs. In *SPAA*, pages 13–22, 2011.
- [6] P. Boldi, M. Rosa, and S. Vigna. HyperANF: approximating the neighbourhood function of very large graphs on a budget. In *WWW*, pages 625–634, 2011.
- [7] S. Chechik, D. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. V. Williams. Better approximation algorithms for the graph diameter. In *SODA*, pages 1041–1052, 2014.
- [8] E. Cohen. Fast algorithms for constructing t -spanners and paths with stretch t . *SIAM J. Comput.*, 28(1):210–236, 1998.
- [9] E. Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *J. ACM*, 47(1):132–166, 2000.
- [10] P. Crescenzi, R. Grossi, M. Habib, L. Lanzi, and A. Marino. On computing the diameter of real-world undirected graphs. *Theor. Comput. Sci.*, 514:84–95, 2013.
- [11] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [12] A. Ene, S. Im, and B. Moseley. Fast clustering using mapreduce. In *KDD*, pages 681–689, 2011.
- [13] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.
- [14] M. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the MapReduce framework. In *ISAAC*, pages 374–383, 2011.
- [15] D. S. Hochbaum and D. B. Shmoys. A best possible parallel approximation algorithm to a graph theoretic problem. *Operational Research*, pages 933–938, 1987.
- [16] U. Kang, C. E. Tsourakakis, A. P. Appel, C. Faloutsos, and J. Leskovec. Hadi: Mining radii of large graphs. *TKDD*, 5(2), 2011.
- [17] D. R. Karger and M. Ruhl. Finding nearest-neighbors in growth-restricted metrics. In *STOC*, pages 741–750, 2002.

- [18] H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *SODA*, pages 938–948, 2010.
- [19] P. N. Klein and S. Sairam. A parallel randomized approximation scheme for shortest paths. In *STOC*, pages 750–758, 1992.
- [20] Laboratory for Web Algorithmics, University of Milano. <http://law.di.unimi.it>
- [21] U. Meyer. On trade-offs in external-memory diameter-approximation. In *SWAT*, pages 426–436, 2008.
- [22] G. L. Miller, R. Peng, and S. C. Xu. Parallel graph decompositions using random shifts. In *SPAA*, pages 196–203, 2013.
- [23] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: a fast and scalable tool for data mining in massive graphs. In *KDD*, pages 81–90, 2002.
- [24] A. Pietracaprina, G. Pucci, M. Riondato, F. Silvestri, and E. Upfal. Space-round tradeoffs for mapreduce computations. In *ICS*, pages 235–244, 2012.
- [25] Stanford Large Network Dataset Collection <http://snap.stanford.edu/data>
- [26] Spark: Lightning-fast cluster computing. <http://spark.apache.org>
- [27] J. D. Ullman and M. Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.*, 20(1):100–125, 1991.
- [28] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.
- [29] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 15–28, 2012.

Table 1: Characteristics of the graphs used in our experiments

Dataset	nodes	edges	diameter
twitter	39,774,960	684,451,342	16
livejournal	3,997,962	34,681,189	21
roads-CA	1,965,206	2,766,607	849
roads-PA	1,088,092	1,541,898	786
roads-TX	1,379,917	1,921,660	1,054
mesh1000	1,000,000	1,998,000	1,998

Table 2: Comparison between the clusterings returned by CLUSTER and MPX. n_C is the number of clusters, m_C is the number of edges between clusters, and r is the maximum cluster radius.

Dataset	Algorithm CLUSTER			Algorithm MPX		
	n_C	m_C	r	n_C	m_C	r
twitter	40001	17216285	5	41431	109348	6
livejournal	4020	230326	7	5796	17098	9
roads-CA	15038	40597	31	16429	34021	61
roads-PA	7710	13300	30	8529	18446	58
roads-TX	10653	28582	30	11238	23308	55
mesh1000	7641	18476	34	9112	25885	56

Table 3: Diameter approximation returned by our algorithm on the benchmark graphs. Δ is the diameter of the graph, Δ' is the approximation given by the algorithm.

Dataset	Coarser clustering				Finer clustering			
	n_C	m_C	Δ'	Δ	n_C	m_C	Δ'	Δ
twitter	1835	18865	23	16	5276	895356	27	16
livejournal	1933	24442	29	21	7837	570608	29	21
roads-CA	1835	5888	1504	849	3863	10946	1477	849
roads-PA	1087	3261	1240	786	4286	12314	1245	786
roads-TX	1316	3625	1568	1054	3821	10880	1603	1054
mesh1000 t	880	3224	2128	1998	3588	14198	2014	1998

Table 4: Comparison of our approach (CLUSTER) with HADI and BFS. Numbers in parentheses are the estimated diameter Δ' . Column Δ reports the original diameter.

Dataset	CLUSTER	Time (Δ')		Δ
		BFS	HADI	
twitter	303 (27)	144 (22)	3697 (14)	16
livejournal	113 (29)	123 (26)	388 (26)	21
roads-CA	742 (1477)	5796 (1418)	11008 (838)	849
roads-PA	369 (1245)	5245 (1244)	10090 (770)	786
roads-TX	622 (1603)	5844 (1466)	12572 (998)	1054
mesh1000	373 (2014)	8627 (2224)	17287 (1998)	1998

Figure 1: Performance of CLUSTER and BFS on graphs with small variations.

